Database Building

ITOS Edition
$Date: 2006/10/04 19:12:30 $

1 Database Overview

In order to ensure that everybody is speaking the same language, here is a brief overview of an ITOS project database.

The database is broken broadly into two parts: telemetry and command.

The telemetry part contains a set of variables, referred to as "mnemonics". There are two broad classes of mnemonics: telemetry and non-telemetry mnemonics some of which are ITOS reserved. Any mnemonic beginning with "GBL_" are reserved for the ITOS software. The values of telemetry mnemonics are sent from the spacecraft telemetry. The values of global mnemonics are set when the database is created and/or are set by the ground system, STOL procedures or configuration monitors running on the ground system.

The telemetry database also contains the maps required to extract the values of telemetry mnemonics from the telemetry stream, the limits on the mnemonic values, and the conversion data required to scale the value or map it to a text string.

The command part of the database contains spacecraft command specifications. Each command has a name, called the "command mnemonic", and consists of a set of fields. Each field is a contiguous set of bits in the telecommand packet.

Each field is named and may be associated with a set of named discrete values. In STOL, the term "submnemonic" is used to refer to a command field: It is either a field name or a name associated with a discrete value from a command discrete value set.

2 Database Exchange Record Overview

ITOS accepts database transactions in ASCII files containing information in the formats layed out below. The database software is able to ingest files in this format.

Records are free format. There may be more than one record per line[1] and one record may span several lines. The sequence '`\nXXX,`' – where '`\n`' is a newline character, '`XXX`' is a ⟨undefined⟩ [record type tag], page ⟨undefined⟩, and '`,`' is a field separater – begins a new record. This means that there need not be a field separater between the end of one record and the start of the next as long as the new record begins at the start of a new line. Spaces or tabs preceeding the record tag are allowed (and ignored).

Records may appear in any order. If there is no information for a particular field, or if the default value is desired, the field may be left blank. Fields missing from the end of a record are presumed blank. Any printable non-alphanumeric character except underscore ('_'), double-quote ('"'), plus sign ('+'), minus sign ('-'), back slash ('\'), space, and pound sign ('#') may be used as a delimiter. Delimiters may be escaped with the backslash ('\') character and will not be interpreted as a delimiter if they appear inside quoted strings.

All names and flags are case insensitive. For things like record tags, mnemonic names, set names, and critical flags, upper and lower case characters are considered identical. Case is preserved only in description and DSC record state text fields. Names must begin with an alphabetic character and may contain only alphabetic and numeric characters and the underscore ('_'). Names typically have length restrictions. The number of characters in names in transaction records are generally unlimited however some constraints do exist in order to keep software structures from getting to large. Also, the user should be aware that PDB files for the CMS and DTAS have fixed menmonic name sizes of 16 characters. So caution should be taken before using name lengths longer than in previous ITOS versions. The following sizes apply:

```
Telemetry records:
    Mnemonic names           - 255 characters.
    Mnemonic units           - 64 characters.
    Mnemonic string values   - unlimited.
    Limit set names          - 255 characters.
    Conversion set names     - 255 characters.
    Discrete value strings   - 255 characters.
    Maximum Packet size      - 65529 bytes.
Command records:
    Mnemonic names           - 255 characters.
    Field names              - 255 characters.
    Submnemonic names        - 255 characters.
    String Submnemonic value - 255 characters.
    Conditional critial exp  - 255 characters.
```

[1] This may change! The side effect of allowing more than one record per line is that '`del`', '`ssi`', '`tlm`' , etc. must be quoted whenever they're not beginning a new record, for example:
dsc,transaction,add,+,1
dsc,transaction,mod,+,1
dsc,transaction,"del",+,1

```
    Checksum routine names   - 255 characters.
    End item verification    - 255 characters.
    Action routines          - 255 characters.
All records:
    Short descriptions       - 64 characters.
    Subsystem list           - 255 characters.
    Subsystems per list      - 16.
```

Note that these records are called "transaction" records. There are no limitations on the numbers of each of these transaction records nor the size of the database generated from them other than physical memory limitations. The database will exist and be maintained in one central location and this database will be considered the "master copy". This master copy will be in the form of a binary image called the ground system operational database.

One of our main goals in the design of these record formats was to avoid dictating to spacecraft engineers what database tools they must use. So we have presented here an ASCII file format that we believe can be generated from common commercial software tools which subsystem engineers might use to maintain parts of the database. They can make database changes in their preferred environment and produce a set of transaction records to apply against the master database, propagating those changes to the ground computer systems.

We understand that not every element of every mission will require all of the functionality these records provide; but, we needed to design them to allow the full range of options and operations of which our ground system is capable.

2.1 The "Record Type" Field

The ITOS database software recognizes thirteen types of transaction records, each of which is prefixed with a three-letter tag describing the contents of the record. The record types are:

'DEL'	change the record field delimiter character
'SSI'	define a subsystem name
'TLM'	a telemetry or global mnemonic definition
'ALG'	an analog conversion coefficient set definition
'DSC'	a discrete conversion state definition
'XPR'	an expression which is used to generate lookup table for analog conversion
'LIM'	a telemetry limit set definition
'MAP'	packet map attributes
'PKT'	a telemetry packet item definition
'SEL'	a packet map selector definition
'CMD'	a command mnemonic definition
'FLD'	a command field definition
'SUB'	a command field discrete value definition

The first field in each record contains one of these three-letter tags indicating the type of information contained in that record.

Note that these three-letter tags are keywords in the database exchange records and must be quoted, for example, if used in Field 3 of a DSC record.

2.2 Key Fields

The subsequent one or more fields contain the identifier or *key* for the record, naming the thing being defined. For example: a mnemonic name is the key for '`tlm`' or '`cmd`' records, and a command mnemonic name with a field name identifies a command field record.

2.3 The "Operation Symbol" Field

The next field of each record contains an operation symbol, either '`+`' or '`-`'. The '`+`' symbol means the record has new or updated information; the '`-`' symbol means the information for the identifier should be deleted from the existing database. Note that deleting a telemetry mnemonic definition will delete all occurrences of that mnemonic in all packets.

Notice that in order to change an identifier, such as a telemetry mnemonic name, you must delete the existing record and add a new record with the same data, but a different name. If you wish to move a mnemonic's value from one source packet to another, you must remember to delete the '`pkt`' entry for the mnemonic in the first packet. If records are not deleted as required, the database will be cluttered with extraneous information and ground system performance may be degraded.

2.4 Numbers and Other Values

2.4.1 Integer and Unsigned Values

Some fields (AppID in a PKT record, for example) require an integer value. Integer values may be specified in decimal, hexadecimal, or binary. (Sorry, they may *not* be specified in octal).

The following all specify the same value:

```
17
017
0x11
0b010001
```

2.4.2 Floating Point Values

Other fields (the limit values in a LIM record, for example) require floating point values. The following are legal floating point values:

```
3.1
.314e1
31.4e-1
-2.1415
```

Note: Integer and Unsigned Values are also valid and will automatically be converted to floating point. All hex values are assumed unsigned.

2.4.3 Time Values

Time values represent a duration of time and are specified in hours, minutes, and seconds. The following are legal time values (all represent 90 minutes):

```
1:30:00
90:00
5400
5400.0
```

2.4.4 Date Values

Date values represent absolute moments in time. Dates are specified in GMT. The following are legal dates:

```
97-12-9:59:59.999
97-12-10:00:00
```

2.4.5 String Values

String values that contain spaces or special characters should be quoted.

2.5 Dates and Times

Date values represent a count of time from some epoch. For historical reasons, the default epoch in ITOS is 00:00:00 on May 24, 1968. Any date telemetry mnemonic or command field may be associated with a different epoch, however, when it is defined in the database.

Alternate epochs are stored in telemetry mnemonics, and the epoch mnemonic is given in the '`field length`' field of the `PKT` or `FLD` record. The actual field length is dictated by the type, and so need not be given for date or time types. The database build program, `dbxodb`, also provides an argument that allows you to specify an epoch mnemonic to be applied to all date and time telemetry mnemonics and command fields that do not otherwise have an epoch mnemonic defined for them. See Chapter 8 [dbxodb], page 37 for details.

Epoch mnemonics must be defined as dates, and should be set to the desired epoch. For example, for a spacecraft counting from the UNIX epoch, set an epoch mnemonic to 70-001-00:00:00. The mnemonic gbl_def_epoch is included in the ITOS DBX inputs as an example: It is set to the default (SMEX) epoch.

Internally to ITOS, all dates are stored relative to the UNIX epoch (a fact which hopefully is transparent to our users), and fractional seconds are stored, in UNIX fashion, as microseconds. The microseconds field of the epoch value contains the conversion factor for transforming microseconds into spacecraft subseconds. So: $(tv\_usec * 10\hat{}-6) * epoch\_usec = sc\_subsec$ where *tv_usec* are microseconds, *epoch_usec* is the epoch microseconds field value, and *sc_subsec* is spacecraft subseconds, and all values are integers. To provide for a time and date values with a subsecond value of 20us per count, for example, set the epoch microseconds field to '`.050000`'. The default is '`.065536`'.

2.6 Type Codes

These are the possible values for

- A TLM record's field 6, the "destination type".
- A PKT record's field 7, the "source type".
- A FLD record's field 5, the "destination type".

Types are:

- **One byte integral types**

  `U1 (UB)`
  `I1 (SB)` These hold signed (`I1`) or unsigned (`U1`) numbers that are no more than 8 bits wide and that don't span octets.

  Signed numbers use two's complement encoding.

  If width is otherwise unspecified in the transaction records, a width of 8 is assumed.

  These may have analog or discrete conversions.

- **Two byte integral types**

  `U12 (UI, UI320)`
  `I12 (SI, SI320)`
  `U21 (UI085)`
  `I21 (SI085)`

  These hold signed (`I12` and `I21`) or unsigned (`U12` and `U21`) numbers that are no more than 16 bits wide and that span two octets. (8-bit or smaller numbers that are contained within one octet could be `U1` or `S1`).

  Two octets get transmitted in big-endian order (high order byte first) for `U12` and `S12` and little-endian order for `U21` and `S21`.

  Signed numbers use two's complement encoding.

  If width is otherwise unspecified in the transaction records, a width of 16 is assumed.

  These may have analog or discrete conversions.

- **Four byte integral types**

  `U1234 (ULI320, U)`
  `I1234 (SLI320, I)`
  `U4321 (ULI085)`
  `I4321 (SLI085)`
  `U3412 (ULI)`
  `I3412 (SLI)`
  `U2143`
  `I2143`
  `UTIM` These hold signed (`I1234`, `I4321`, `I3412`, and `I2143`) or unsigned (`U1234`, `U4321`, `U3412`, and `U2143`) numbers that are up to 32 bits wide and that span at least three octets (values that span only one or two octets could be one of the smaller integral types).

Four octets get transmitted: for `U1234`, `I1234`, and `UTIM` the octets get transmitted in big-endian order (high order byte first); for `U4321` and `I4321` the octets get transmitted low order byte first; for `U3412` and `I3412` the octets get transmitted in the bizarre order next-to-low order byte first, low order byte second, high order byte third, and next-to-high order byte last, and for `U3412` and `I3412` the octets get transmitted in the equally bizarre order next-to-high order byte first, high order byte second, low order byte third, and next-to-low order byte last.

Signed numbers use two's complement encoding.

If width is otherwise unspecified in the transaction records, a width of 32 is assumed.

Except for `UTIM`, these may have analog or discrete conversions. `UTIM` implicitly gets converted to a date: A `UTIM`'s value is the difference between the computer's idea of the current date and some desired date; the desired date then is the current value.

- **Four byte floating point**

`F1234 (SFP320)`
`F3412 (SFP)`
`F4321`
`F2143`

32-bit IEEE-754 floating point numbers.

Four octets get transmitted: for `F1234` the octets get transmitted in big-endian order (high order byte first); for `F4321` the octets get transmitted low order byte first; for `F3412` the octets get transmitted in the bizarre order next-to-low order byte first, low order byte second, high order byte third, and next-to-high order byte last, and for `F2143` the octets get transmitted in the equally bizarre order next-to-high order byte first, high order byte second, low order byte third, and next-to-low order byte last.

The only legal width is 32.

These may have analog or discrete conversions.

- **Eight byte floating point**

`F12345678 (DFP320)`
`F78563412 (DFP, F)`
`F87654321`
`F43218765`
`F21436587`

64-bit IEEE-754 floating point numbers.

Eight octets get transmitted: for `F12345678` the octets get transmitted in big-endian order; for `F78563412` the next-to-low order byte is transmitted first and the next-to-high order byte last; for `F87654321` the octets get transmitted in strict little-endian order; for `F43218765` the octets get transmitted in a kind of 'little-endian with long-words swapped' order; and for `F21436587` the next-to-high order byte is transmitted first and the next-to-low order byte last.

If width is otherwise unspecified in the transaction records, a width of 64 is assumed. (In fact, the only legal width is 64).

These may have analog or discrete conversions.

- **BCD floating point**

`B12345678 (DFP085, B)`

8-byte BCD encoded number. These are obsolete and should no longer be used.

The first byte of an `B12345678` contains the sign and the base-10 exponent. The remaining seven bytes contain the 14 BCD digit mantissa. `B12345678` values are transmitted in big-endian order.

The sign bit is the high order bit of the first byte; `0` means the value is positive, `1` means the value is negative.

The base-10 exponent is determined by subtracting `64` from the value of the low order seven bits in the first byte; the exponent ranges from `-64` through `63`.

`B12345678` numbers are normalized so that the high order digit of the mantissa is non-zero unless the value is zero.

`B12345678` values range from `-9.9999999999999e63` through `-1.0000000000000e-64`, `0`, `1.0000000000000e-64` to `9.9999999999999e63`.

Some examples of `B12345678` numbers:

```
1      ⇒ 0x4010000000000000
1e-64  ⇒ 0x0010000000000000
0      ⇒ 0x0000000000000000
-1     ⇒ 0xC010000000000000
-9.8765432101234e27 ⇒ 0xDB98765432101234
```

- **CUC time**

`RTIME12` A 3-octet relative time. The first octet (or byte) is the number of seconds, the second and third octets form a 16 bit little-endian number (I.e. the third byte is the high order byte) that represents the number of fractional (2^-16) seconds.

`RTIME20` A 2-octet relative time. The octets are transmitted in little-endian order; the two octets form a 16 bit number of seconds.

`RTIME40`
`RTIME42` A 4-octet (`RTIME40`) or 6-octet (`RTIME42`) absolute time. The first four octets form a 32 bit little-endian (i.e. I4321) number of seconds that represents the moment in time that many seconds after the epoch. For `RTIME42`, the last two octets format a 16 bit little-endian number that represents the number of fractional (2^-16) seconds.

`TIME12` A 3-octet relative time. The octets are transmitted in big-endian order; the first octet (or byte) is the number of seconds, the second and third octets form a 16 bit number that represents the number of fractional (2^-16) seconds.

`TIME20` A 2-octet relative time. The octets are transmitted in big-endian order; the two octets form a 16 bit number of seconds.

`TIME40`
`TIME42` A 4-octet (`TIME40`) or 6-octet (`TIME42`) absolute time. The octets are transmitted in big-endian order; the [first] four octets form a 32 bit number of seconds that represents the moment in Time that many seconds after the epoch. For `TIME42` the last two octets format a 16 bit number that represents the number of fractional (2^-16) seconds.

- **PB1 time** *(a.k.a. NASA-36)*

`PB1` A 6-octet (48 bit) big-endian absolute time. The first 12 bits are ignored, the next 9 bits are the day of year, and the final 27 bits are milliseconds of day. Since PB1 has no fixed epoch it is up to the user to set up an epoch of Jan 1 of the current year or the date won't come out right. *This format has very limited usefulness and is not supported for spacecraft commanding which will generate an ITOS runtime panic event if used in a CMD definition record.*

- **PB5 time**

`TIME`
`TTIM` A 7-octet (`TIME`) or 6-octet (`TTIM`) big-endian absolute time. The first two octets form a 16 bit number of days since the epoch. The third, fourth and fifth octets form the 24 bit seconds of the day. The last 2 (`TIME`) octets form a 16 bit thousanths of the second or 1 (`TTIM`) octet forms a an 8 bit hundreths of a second.

- **non-CUC time**

`RTIMET42` A 6-octet absolute time. The first four octets form a 32 bit little-endian (i.e. U4321) number of 1/10 seconds that represents the moment in time that many seconds after the epoch. The last two octets format a 16 bit little-endian number that represents the number of fractional (2^-16) seconds. Rollover of the date will occur about 13 years 203 days past the epoch. Because the seconds are unsigned there can be no dates prior to the epoch. Subseconds are not a function of the epoch but are fixed at 2 usec granularity.

`TIMET42` A 6-octet absolute time. The first four octets form a 32 bit big-endian (i.e. U1234) number of 1/10 seconds that represents the moment in time that many seconds after the epoch. The last two octets format a 16 bit little-endian number that represents the number of fractional (2^-16) seconds. Rollover of the date will occur about 13 years 203 days past the epoch. Because the seconds are unsigned there can be no dates prior to the epoch. Subseconds are not a function of the epoch but are fixed at 2 usec granularity.

`RTIME44` A 8-octet absolute time. The first four octets form a 32 bit little-endian (i.e. I4321) number of seconds that represents the moment in time that

many seconds after the epoch. The last four octets format a 32 bit little-endian number that can represent the number of fractional seconds up to 2^-32. The default fractional tick if not specified in the epoch mnemonic is (1e-6) one microsecond.

`TIME44` A 8-octet absolute time. The first four octets form a 32 bit big-endian (i.e. I1234) number of seconds that represents the moment in time that many seconds after the epoch. The last four octets format a 32 bit big-endian number that can represent the number of fractional seconds up to 2^-32. The default fractional tick if not specified in the epoch mnemonic is (1e-6) one microsecond.

- **String**

`S1 (CHAR, S)`
Character string that gets transmitted as bytes, so there are no endianness issues.

`S21` Character strings that get transmitted in 16-bit little-endian order; the string `"Hi mom"` gets transmitted as `"iHm mo"`. Note that the length of these strings must be even!

2.7 Comments

Comments begin with a '`#`' character and continue to the end of the line. '`\#`' is a quoted '`#`' that does not begin a comment; this provides a way to get '`#`' in description fields (another way is to quote the description). When a comment occurs in the middle of a record, the comment is treated as white space. For example, the following is allowed:

```
ALG|CNVSET1|+|        # Analog conversion CNVSET1:
             | -3.6065400e+01  # C[0]
             | 1.78768000e-01  # + C[1] X
             | -5.9817700e-04  # + C[2] X^2
```

2.8 Quoted Values

The information provided for any field may appear inside double quotes. Delimiter and comment characters are not interpreted when they appear in a quoted string.

Double quoted strings may span several lines.

2.9 Description Fields

Most records contain a "description" field, which may contain either or both of the item's long description and the item's short description. If the description field contains the string '`<HTML>`', everything before '`<HTML>`' is the short description and '`<HTML>`' begins the long description. Otherwise, there is no long description. Short descriptions may get truncated if longer than 62 characters. Short descriptions are used in event messages, interactive prompts, and printed reports.

Should the description field be longer than 62 characters and contain no '<HTML>' tag, the description will be split on the nearest word boundary less than 63 characters into the short description and the remainder placed in the long description.

As implied by '<HTML>', long descriptions are written in HTML and are intended to be read using a Word Wide Web browser. Long descriptions may include pictures and may have hypertext links to other descriptions or documents.

The following shows a transaction record with both short and long descriptions:

```
TLM,GBL_PROCPATH,+,,ITOS,CHAR,999,,,,,,".",T,
    "List of directories STOL searches when looking for procs.
     <html>The colon-seperated list of directories, modeled after the
     Unix environment variable <var>PATH</var>, that
     <font size=-1>STOL</font>
     searches when looking for a proc.
     <p>
     If the proc occurs in more than one of the directories in
     <var>GBL_PROCPATH</var>, the proc from the first of those
     directories is used.
     <p>
     See also <a href=\"GBL_PAGEPATH.html\">GBL_PATHPATH</a>."
```

2.9.1 Using relative URLs to reference other descriptions

Long descriptions allow items reference each other (and other ITOS documents) using relative URLs. Long descriptions are inserted into machine-generate html files which are organized as follows:

htmldir`/dir.html`
: Machine generated table of contents.

htmldir`/packets`
: Directory containing machine generated descriptions of the packet maps. Files are named using '`"app%04d.html",appid`' where '`appid`' is the packet's appid number: `app0001.html` contains the machine generated description of the appid 1 packet. Long descriptions from PKT records may be inserted in these files.

htmldir`/mnemonics`
: Directory containing machine generated descriptions of telemetry mnemonics. Files are named using '`"%s.html",mnemonic`' where '`mnemonic`' is the uppercase mnemonic name: `SCCMDCNT.html` contains the machine generated description of mnemonic SCCMDCNT. Long descriptions from TLM, LIM, ALG, and DSC records may be inserted in these files.

htmldir`/commands`
: Directory containing machine generated descriptions of commands. Files are named using '`"%s.html",command`' where '`command`' is the uppercase name of the command: `SNOOP.html` contains the machine generated description of command SNOOP. Note that command names are uppercase! Long descriptions from CMD, FLD, and SUB records may be inserted in these files.

htmldir`/subsystems`

Directory containing machine generated descriptions of subsystems. Files are named using '`"%s.html",subsystem`' where '`subsystem`' is the uppercase subsystem name: `ACS.html` contains the machine generated description of the ACS subsystem. Long descriptions from SSI records are inserted in these files.

2.9.2 Using relative URLs to reference other ITOS documents

The ITOS documentation is assumed to be installed in a directory named '`ITOS`' in the same directory as *htmldir*. So, to access the documentation for the START directive from the description of global mnemonic GBL_PROCPATH, use

```
... <a href=\"../../ITOS/directives/START.html\">START</a> ...
```

(You have to use backslash-quote (\") since this is a description field).

See section "WWW Document Organization" in *ITOS Documentation Overview*, for discussion of how the ITOS documentation is arranged.

3 General Records

3.1 DEL

The '`del`' record is provided to allow users to change the record field delimiter from the default comma ('`,`') to another character.

Field 1 '`DEL`', indicating the record type.

Field 2 The new field delimiter; any printable non-alphanumeric character except underscore ('`_`'), double-quote ('`"`'), plus sign ('`+`'), minus sign ('`-`'), back slash ('\'), space, and pound sign ('`#`').

The following changes the field delimiter to '`|`':

```
DEL,|
```

3.2 SSI

Telemetry and command mnemonics and telemetry packets may be assigned to one or more subsystems. The subsystem name may be used to limit database searches or reports, or simply for information. It also may be used to grant command permission to send spacecraft commands on a per-subsystem basis for ground systems which allow commands to be sent from multiple workstations.

Field 1 '`SSI`', indicating the record type.

Field 2 The "subsystem name". The acronym or name that identifies the subsystem. Subsystem names must begin with a letter, may contain only alphanumeric characters or underscore ('`_`'), and may be up to 255 characters long.

Field 3 '`+`' or '`-`', indicating the operation symbol.

Field 4 Subsystem description.

The following defines the ACS subsystem:

```
SSI,acs,+,Attitude Control and Stabilization
```

4 Telemetry Records

The basic definition sequence of a telemetry stream starts with the definition of a MAP record for a specific segment of the stream. In CCSDS this is defined by the application id (AppID). A MAP record is defined for each AppID. The MAP record defines general information about a given AppID stream referred to as the packet attributes.

Next, a series of PKT records that go with the MAP record for a given AppID are defined. Each PKT record defines one piece of the total telemetry packet and specifies where to extract a value from the packet and what TLM mnemonic record to use to store the result. Although it not necessary, it is a good habit by convention to group the MAP and all the PKT records for a single AppID together for ease of editing and fixing errors. It is also wise to order the PKT records in ascending order by offset in the packet so it is easy to see missing or overlapping records.

The TLM records define where a value extracted from a packet is stored in ITOS's realtime value tables. One TLM record can be referenced in multiple PKT records because a data item can come down in a stream multiple times but it is only stored in one place. The database compiler '`DBXODB`' will warn of the use the same TLM record in multiple PKT records because it is an unusual case so that the user is sure it was not done by mistake.

Along with the TLM mnemonic record, optional limit and conversion records that are specified in the PKT record must be defined. It is not necessary to define unique limit and conversion records for each PKT because they can be shared and reused in any PKT record for any AppID.

LIMITS are defined by using the LIM record. Limit records define upper and lower bounds of a number to determine a YELLOW and RED state or limit. These limits will trigger events messages when the thresholds are crossed in two consecutive packets.

Conversion of the PKT data is defined by two types, ANALOG which use the ALG records and DISCRETE which use the DSC records. An ANALOG conversion specifies how to convert analog value to engineering units by applying a polynomial equation. A DISCRETE conversion specifies how to take a numeric value and convert it to a text string.

Here is the simple example packet definition for AppID 5:

```
DEL,|
SSI | Packet_Overhead |+| Telemetry Packet Fields          |
SSI | DS_telemetry    |+| Data Storage Subsystem Telemetry |

MAP | 5 |+| DS_telemetry   |||H005TIME| "APID 5, Data Storage Event Packet" |
PKT | 5 | H005PVNO    | 0 |+||     UB |  0 | 0 |  3 | 0 |||
PKT | 5 | H005PCKT    | 0 |+||     UB |  0 | 3 |  1 | 0 |||
PKT | 5 | H005SHDF    | 0 |+||     UB |  0 | 4 |  1 | 0 |||
PKT | 5 | H005APID    | 0 |+||     UI |  0 | 5 | 11 | 0 |||
PKT | 5 | H005SEGF    | 0 |+||     UB |  2 | 0 |  2 | 0 |||
PKT | 5 | H005CNT     | 0 |+||     UI |  2 | 2 | 14 | 0 |||
PKT | 5 | H005PLEN    | 0 |+||     UI |  4 | 0 | 16 | 0 |||
PKT | 5 | H005SECH    | 0 |+||     UI |  6 | 0 | 16 | 0 |||
PKT | 5 | H005TIME    | 0 |+|| TIME42 |  6 | 0 | 48 | 0 |||
PKT | 5 | H005SECL    | 0 |+||     UI |  8 | 0 | 16 | 0 |||
```

```
PKT | 5 | H005SUBS    | 0 |+||     UI | 10 | 0 | 16 | 0 |||
PKT | 5 | DSEVENTSSYS | 0 |+||  U1234 | 12 | 0 | 32 | 0 |||
PKT | 5 | DSEVENTSEV  | 0 |+||     UB | 16 | 0 |  8 | 0 |||
PKT | 5 | DSEVENTHUB  | 0 |+||     UB | 17 | 0 |  8 | 0 |||
PKT | 5 | DSEVENTCODE | 0 |+||     UI | 18 | 0 | 16 | 0 |||
PKT | 5 | DSEVENTDAT1 | 0 |+||  U1234 | 20 | 0 | 32 | 0 |||
PKT | 5 | DSEVENTDAT2 | 0 |+||  U1234 | 24 | 0 | 32 | 0 |||
PKT | 5 | DSEVENTDAT3 | 0 |+||  U1234 | 28 | 0 | 32 | 0 |||
PKT | 5 | DSEVENTDAT4 | 0 |+||  U1234 | 32 | 0 | 32 | 0 |||
PKT | 5 | DSEVENTSTR  | 0 |+||     S1 | 36 | 0 | 40 | 0 |||

TLM | H005PVNO    |+|| Packet_Overhead |     UB |    ||   | 1 ||                    | 0
TLM | H005PCKT    |+|| Packet_Overhead |     UB |    ||   | 1 ||                    | 0
TLM | H005SHDF    |+|| Packet_Overhead |     UB |    ||   | 1 ||                    | 0
TLM | H005APID    |+|| Packet_Overhead |     UI |    ||   | 1 ||                    | 0
TLM | H005SEGF    |+|| Packet_Overhead |     UB |    ||   | 1 ||                    | 0
TLM | H005CNT     |+|| Packet_Overhead |     UI |    ||   | 1 ||                    | 0
TLM | H005PLEN    |+|| Packet_Overhead |     UI |    ||   | 1 ||                    | 0
TLM | H005SECH    |+|| Packet_Overhead |     UI |    ||   | 1 ||                    | 0
TLM | H005TIME    |+|| Packet_Overhead | TIME42 |    ||   | 1 ||                    |
TLM | H005SECL    |+|| Packet_Overhead |     UI |    ||   | 1 ||                    | 0
TLM | H005SUBS    |+|| Packet_Overhead |     UI |    ||   | 1 ||                    | 0
TLM | DSEVENTSSYS |+|| DS_telemetry    |  U1234 |    || S | 1 || EVT_SUBSYS_DSC    | 0
TLM | DSEVENTSEV  |+|| DS_telemetry    |     UB |    || R | 1 || EVT_SEVERITY_DSC | 0
TLM | DSEVENTHUB  |+|| DS_telemetry    |     UB |    ||   | 1 || EVT_HUB_ID_DSC    | 0
TLM | DSEVENTCODE |+|| DS_telemetry    |     UI |    || C | 1 || EVT_CODE_DSC      | 0
TLM | DSEVENTDAT1 |+|| DS_telemetry    |  U1234 |    || 1 | 1 ||                    | 0
TLM | DSEVENTDAT2 |+|| DS_telemetry    |  U1234 |    || 2 | 1 ||                    | 0
TLM | DSEVENTDAT3 |+|| DS_telemetry    |  U1234 |    || 3 | 1 ||                    | 0
TLM | DSEVENTDAT4 |+|| DS_telemetry    |  U1234 |    || 4 | 1 ||                    | 0
TLM | DSEVENTSTR  |+|| DS_telemetry    |     S1 | 40 ||   | 1 ||                    | 0

DSC |EVT_CODE_DSC |"DS Initialized with Recorder Data Intact ( DEVICE )"             |
DSC |EVT_CODE_DSC |"DS Initialized with Recorder Reinitialized ( DEVICE )"           |
DSC |EVT_CODE_DSC |"Storage Disk Creation Failed ( DEVICE )"                         |
DSC |EVT_CODE_DSC |"Invalid Storage Mode Parameter Value ( MODE )"                   |
DSC |EVT_CODE_DSC |"Recorder File Full ( DEVICE )"                                   |
DSC |EVT_CODE_DSC |"NEW Data Group Overwrite Detected ( DEVICE )"                    |
DSC |EVT_CODE_DSC |"OLD Data Group Overwrite Detected ( DEVICE )"                    |
DSC |EVT_CODE_DSC |"Storage Device Name Not Present"                                 |
DSC |EVT_CODE_DSC |"Storage Device Id Match Not Found"                               |
DSC |EVT_CODE_DSC |"Storage Device For System Events Not Found"                      |
DSC |EVT_CODE_DSC |"Invalid Map Table Item ( ITEM| FIELD| REASON )"      |+| 510.0 | 5
DSC |EVT_CODE_DSC |"Map Table Invalidated Due To Error"      |+| 511.0 | 511.0 | WHITE
DSC |EVT_CODE_DSC |"Invalid Data Group Parameter Value"      |+| 512.0 | 512.0 | WHITE
DSC |EVT_CODE_DSC |"Map Table Specified More Than One Event Packet Storage Device"   |
DSC |EVT_CODE_DSC |"Devices Defined in Map Overlap ( DEVICE| DEVICE )"      |+| 514.0
DSC |EVT_CODE_DSC |"Invalid Filename Command Parameter"      |+| 515.0 | 515.0 | WHITE
```

```
DSC |EVT_CODE_DSC |"DS Initialized with Recorder Data Intact ( DEVICE )"            |
DSC |EVT_CODE_DSC |"DS Initialized with Recorder Reinitialized ( DEVICE )"          |
DSC |EVT_CODE_DSC |"Storage Disk Creation Failed ( DEVICE )"                        |
DSC |EVT_CODE_DSC |"Invalid Storage Mode Parameter Value ( MODE )"                  |
DSC |EVT_CODE_DSC |"Recorder File Full ( DEVICE )"                                  |
DSC |EVT_CODE_DSC |"NEW Data Group Overwrite Detected ( DEVICE )"                   |
DSC |EVT_CODE_DSC |"OLD Data Group Overwrite Detected ( DEVICE )"                   |
DSC |EVT_CODE_DSC |"Storage Device Name Not Present"                                |
DSC |EVT_CODE_DSC |"Storage Device Id Match Not Found"                              |
DSC |EVT_CODE_DSC |"Storage Device For System Events Not Found"                     |
DSC |EVT_CODE_DSC |"Invalid Map Table Item ( ITEM| FIELD| REASON )"      |+| 510.0 | 5
DSC |EVT_CODE_DSC |"Map Table Invalidated Due To Error"      |+| 511.0 | 511.0 | WHITE
DSC |EVT_CODE_DSC |"Invalid Data Group Parameter Value"      |+| 512.0 | 512.0 | WHITE
DSC |EVT_CODE_DSC |"Map Table Specified More Than One Event Packet Storage Device"   |
DSC |EVT_CODE_DSC |"Devices Defined in Map Overlap ( DEVICE| DEVICE )"      |+| 514.0
DSC |EVT_CODE_DSC |"Invalid Filename Command Parameter"      |+| 515.0 | 515.0 | WHITE

DSC |EVT_HUB_ID_DSC   |"????"      |+|  0.0 |   0.0 | WHITE | BLACK ||
DSC |EVT_HUB_ID_DSC   |"COMP"      |+|  1.0 |   1.0 | WHITE | BLACK ||
DSC |EVT_HUB_ID_DSC   |"EPIC"      |+|  2.0 |   2.0 | WHITE | BLACK ||
DSC |EVT_HUB_ID_DSC   |"????"      |+|  3.0 | 255.0 | WHITE | BLACK ||

DSC |EVT_SEVERITY_DSC |"UNDEFINED" |+|  0.0 |   0.0 | WHITE | BLACK ||
DSC |EVT_SEVERITY_DSC |"INFO"      |+|  1.0 |   1.0 | WHITE | BLACK ||
DSC |EVT_SEVERITY_DSC |"OPERROR"   |+|  2.0 |   2.0 | WHITE | BLACK ||
DSC |EVT_SEVERITY_DSC |"SWWARN"    |+|  3.0 |   3.0 | WHITE | BLACK ||
DSC |EVT_SEVERITY_DSC |"HWWARN"    |+|  4.0 |   4.0 | WHITE | BLACK ||
DSC |EVT_SEVERITY_DSC |"SWERROR"   |+|  5.0 |   5.0 | WHITE | BLACK ||
DSC |EVT_SEVERITY_DSC |"HWERROR"   |+|  6.0 |   6.0 | WHITE | BLACK ||
DSC |EVT_SEVERITY_DSC |"UNDEFINED" |+|  7.0 | 255.0 | WHITE | BLACK ||

DSC |EVT_SUBSYS_DSC   |"OS"        |+|  0.0 |   0.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"BG"        |+|  1.0 |   1.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"SB"        |+|  2.0 |   2.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"XI"        |+|  3.0 |   3.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"XO"        |+|  4.0 |   4.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"SH"        |+|  5.0 |   5.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"HK"        |+|  6.0 |   6.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"CI"        |+|  7.0 |   7.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"TO"        |+|  8.0 |   8.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"ST"        |+|  9.0 |   9.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"SM"        |+| 10.0 |  10.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"LC"        |+| 11.0 |  11.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"DS"        |+| 12.0 |  12.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"SC"        |+| 13.0 |  13.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"PM"        |+| 14.0 |  14.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"MS"        |+| 15.0 |  15.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"CS"        |+| 16.0 |  16.0 | WHITE | BLACK ||
```

```
DSC |EVT_SUBSYS_DSC   |"AC"        |+| 17.0 |  17.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"AM"        |+| 18.0 |  18.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"TM"        |+| 19.0 |  19.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"VS"        |+| 20.0 |  20.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"PN"        |+| 21.0 |  21.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"SE"        |+| 22.0 |  22.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"ID"        |+| 23.0 |  23.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"FM"        |+| 24.0 |  24.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"EC"        |+| 25.0 |  25.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"ED"        |+| 26.0 |  26.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"EH"        |+| 27.0 |  27.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"XF"        |+| 28.0 |  28.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"EX"        |+| 29.0 |  29.0 | WHITE | BLACK ||
DSC |EVT_SUBSYS_DSC   |"??"        |+| 30.0 | 255.0 | WHITE | BLACK ||
```

The following is a telemetry mnemonic definition for an ITOS system variable. It is included here because it contains a nice bit of HTML documentation. This mnemonic doesn't have an associated PKT record because it doesn't come down in telemetry. You might create something like this to hold pseudo-telemetry produced by the ITOS configuration monitor.

```
TLM |GBL_CM_TXPORT |+||ITOS_CMD |S1|32|||1|||CLIENT_TCP|F|
    "Command destination host connection type.
    <html>Determines whether ITOS initiates a TCP/IP connection,
    accepts a TCP/IP connection, or makes a UDP/IP association for
    spacecraft commanding.  For security reasons, ground stations may
    want to initiate the TCP connection.
    <p>Values are:
    <dl>
    <dt><code>client_tcp</code>
    <dd> ITOS initiates a connection from an arbitrary port on the
         localhost to port <a href=\"GBL_CMD_PORT.html\">
         GBL_CMD_PORT</a> on host <a href=\"GBL_CMD_HOST.html\">
         GBL_CMD_HOST</a>.
    <dt><code>server_tcp</code>
    <dd> ITOS listens for a connection from any host and port on port
         GBL_CMD_PORT.
    <dt><code>udp</code>
    <dd> ITOS makes a UDP assocation using the same host and port
         configuration as the <code>client_tcp</code> mode.  If
         GBL_CMD_HOST resolves to a multicast (class D) address, the
         multicast time-to-live is set automatically to 127.
    <dt><code>serial</code>
    <dd> ITOS opens the serial port (RS-423/RS-232) given by <a
    href=\"GBL_CMD_FILE.html\">GBL_CMD_FILE</a> and configures it
         according to <a
         href=\"GBL_CMD_SERIAL.html\">GBL_CMD_SERIAL</a>.
    <dt><code>file</code>
    <dd> ITOS sends commands to the file given by <a
```

```
        href=\"GBL_CMD_FILE.html\">GBL_CMD_FILE</a>.
    </dl>"
```

See *GBL_CM_TXPORT* to view the HTML output that is generated from the above.

The telemetry records are

'TLM' define a telemetry mnemonic,

'ALG' define an analog conversion,

'DSC' define a discrete conversion,

'LIM' define a limit set,

'MAP' specify packet map attributes,

'PKT' define an item in a packet map, and

'SEL' define a packet map selector.

4.1 The TLM Record

The 'TLM' record defines telemetry and global mnemonics.

Note that because mnemonics need not be in the telemetry stream, the definition of a mnemonic does _not_ include where in the telemetry stream that mnemonic may be found. Non-telemetry mnemonics may be ITOS system state variables (a so-called "global mnemonic" that begins with "GBL_") or a user-defined mnemonic created to hold a derived value or a constant value. Mnemonics also may appear multiple times in the telemetry stream, in the same packet or in different packets.

Field 1 'TLM', indicating the record type.

Field 2 The telemetry "mnemonic" is the name of the telemetry data point or ground system global value. The value is accessed through this name, just like a variable in a program (which it essentially is in the case of STOL proceedures). Mnemonic names must begin with a letter, may contain only alphanumeric characters or underscore ('_'), and may be up to 255 characters long. Mnemonics that begin with "GBL_" are reserved for system use and the user should refrain from using them.

Field 3 '+' or '-', indicating the operation symbol.

Field 4 Mnemonic ID. This is a historical artifact from the FAST mission and is normally left blank. For FAST, this was a unique integral number that identified the mnemonic between 0 and 65534. A blank ID defaults to 65535.

Field 5 List of subsystem names (see Section 3.2 [ssi], page 13) indicating which subsystems this mnemonic is associated with. Up to 16 subsystem names may be specified using spaces to seperate the names.

Field 6 The "destination type" of the mnemonic. Indicates how the mnemonic's value is stored and used within the ITOS. Also used as the default "source type" if none is specified in the PKT record.

Field 7 The size of the mnemonic's value, in bits or, if the "type" is a string type, in bytes. If blank, this field value will be inferred from the type, if possible. For example, a `U12` defaults to 16 bits. Signed and unsigned integers may not be larger than 32 bits; floating-point values may be 32 or 64 bits; CUC times and dates may be 16, 24, 32, or 48 bits; BCD values must be 64 bits; and PB5 times are 48 or 56 bits; and strings have no default size. Note that the size given here is used to limit the range of values that STOL may assign to a mnemonic. For example, a size of 2 bits limits an unsigned integer mnemonic's values to the range 0 to 3.

Field 8 Engineering units string (up to 64 chars). Used when displaying the mnemonic's value.

Field 9 The event flag is a single alphanumeric character, used to mark mnemonics which are used to telemeter spacecraft events. This field was first used for SWAS, where event packets contained an event code, a subsystem identifier, a severity code, and four datapoints.

C	Event code
S	Subsystem ID
R	Severity
1	Data Point 1
2	Data Point 2
3	Data Point 3
4	Data Point 4

(FAST handled events differently, using event codes '`C`', '`I`', and '`V`'. The FAST method is deprecated and only mentioned here as a reminder that it did things differently). The default is ' '.

Field 10 Array length (default = 1).

Field 11 Limits. This field may contain a name, a number, or a name and a number seperated by whitespace.

If the field contains a name, that name is the limit definition (see Section 4.5 [lim], page 22) that specifies the red/yellow limits for this mnemonic. (If the field does not contain a name, the mnemonic has no red/yellow limits).

If the field contains a number, that number is the delta limit for the mnemonic.

Field 12 The name of the analog (see Section 4.2 [alg], page 20), expression (see Section 4.4 [xpr], page 21), or discrete (see Section 4.3 [dsc], page 20) conversion to apply to this mnemonic value when displaying the value or if it is referenced through the STOL '`p@`' operator. If blank, the mnemonic has no conversion.

Field 13 The mnemonic's initial value. If blank, the mnemonic in unitialized.

Field 14 '`T`' or '`F`'. If '`T`', normal STOL assignment directives will not be permitted to change this mnemonic's value. The default is '`F`'.

Field 15 The mnemonic description.

4.2 The ALG Record

Analog conversions, defined in ALG records, are intended for converting an integer number of "counts" (the output of an analog to digital converter, for example) into a floating-point value in "engineering units", such as volts, amps, degrees, etc. The ALG record defines the coefficients for an 8th order polynomial. The integer or floating-point telemetry value is applied to the polynomial and the result is a floating-point value.

Field 1 'ALG', indicating the record type.

Field 2 The analog conversion definition name, which is used in the 'tlm' record's conversion definition name. Conversion names must begin with a letter, may contain only alphanumeric characters or underscore ('_'), and may be up to 255 characters long. Analog, discrete, and expression conversions may not have the same name.

Field 3 '+' or '-', indicating the operation symbol.

Field 4 A floating point value representing C0 in the polynomial
C0 + C1*X + C2*X^2 + ... + C7*X^7
The default is '0.0'.

Field 5 A floating point value representing C1, the coefficient of X^1. The default is '0.0'.

Field 6 A floating point value representing C2, the coefficient of X^2. The default is '0.0'.

Field 7 A floating point value representing C3, the coefficient of X^3. The default is '0.0'.

Field 8 A floating point value representing C4, the coefficient of X^4. The default is '0.0'.

Field 9 A floating point value representing C5, the coefficient of X^5. The default is '0.0'.

Field 10 A floating point value representing C6, the coefficient of X^6. The default is '0.0'.

Field 11 A floating point value representing C7, the coefficient of X^7. The default is '0.0'.

Field 12 The analog conversion's description.

4.3 The DSC Record

Discrete conversions, defined in DSC records, transform a range of numeric values into a set of text strings. The telemetry value is compared to each range in the set. If the value falls within a range, the state text associated with that range is displayed. The high value of one range can be the same as the low value of the next range; otherwise overlapping ranges are discouraged.

Field 1 'DSC', indicating the record type.

Field 2 The discrete conversion definition name, which is used in the 'tlm' record's conversion definition name (field 12). Conversion names must begin with a letter, may contain only alphanumeric characters or underscore ('_'), and may be up to 255 characters long. Discrete and analog, and expression conversions may not have the same name.

Field 3 The state text. This is the string displayed if the mnemonic value falls in the given range. This string may contain non-alphanumeric characters (in which case it should be quoted). There is no max length of any one state text but however the strings are truncated after 255 characters.. .

Field 4 '+' or '-', indicating the operation symbol.

Field 5 A floating point value representing the low value for the range. The default is '-DBL_MAX'.

Field 6 A floating point value representing the high value for the range. The default is 'DBL_MAX'.

Values are compared with the discrete range as:

```
low <= value <= high
```

Field 7 An integer value or name indicating the foreground (text) color. The default is '7' (white). The color names are:

black – 0

red – 1

green – 2

yellow – 3

blue – 4

magenta – 5

cyan – 6

white – 7

Field 8 An integer value or name (see Field 7 above) indicating the background color. The default is '0' (black).

Field 9 The discrete conversion's description.

4.4 The XPR Record

The XPR record defines a STOL-language expression of one variable which is used to generate a look-up table.

XPR,name,+,expression,input_bits,signed?,description

defined in XPR records, are intended for converting an integer number of "counts" (the output of an analog to digital converter, for example) into a floating-point value in

"engineering units", such as volts, amps, degrees, etc. The XPR record defines an expression of one variable which is used to build a lookup table. The integer (signed or unsigned) telemetry value is used as an index to the table and the result is a floating-point value.

Field 1 'XPR', indicating the record type.

Field 2 The expression name, which is used in the 'tlm' record's conversion definition name field (field 12). Conversion names must begin with a letter, may contain only alphanumeric characters or underscore ('_'), and may be up to 255 characters long. Discrete, analog, and expression conversions may not have the same name.

Field 3 '+' or '-', indicating the operation symbol.

Field 4 A STOL-language expression of one variable, *x*, which is used to generate a lookup table of 2^n values, where n is given in field 5. The values in the table are stored as 64-bit IEEE floating-point values. For example:

log(x) / .25 3 + 2.3*(x+10) + 3.021*(x+10)^2 + .002*(x+10)^4

Field 5 The number of bits in the input value, *x*.

Field 6 The signed flag, which may be 'T' or 'F'(default). If 'T', the value plugged into the variable in the expression is signed. So, for example, if the input value is 8 bits, and the flag is true, all values between -128 and 127 are fed through the expression. The result for each *x* is stored in the table at index `abs(x)`.

Field 7 The expression / table description.

4.5 The LIM Record

The LIM record defines limits for an integer or floating-point telemetry mnemonic. A limit set consists of two concentric ranges called the "yellow limits" (fields 5 & 6) and "red limits" (fields 4 & 7). A limit definition may contain more than one limit set. The system chooses which limit set to apply to a mnemonic using the "limit switch", explained below.

Field 1 'LIM', indicating the record type.

Field 2 The limit definition name, which is used in the 'tlm' record's limit field (field 11). Limit definition names must begin with a letter, may contain only alphanumeric characters or underscore ('_'), and may be up to 255 characters long.

Field 3 '+' or '-', indicating the operation symbol.

Field 4 A floating point value representing the red low limit. The default is '-DBL_MAX'.

Field 5 A floating point value representing the yellow low limit. The default is '-DBL_MAX'.

Field 6 A floating point value representing the yellow high limit. The default is 'DBL_MAX'.

Field 7 A floating point value representing the red high limit. The default is 'DBL_MAX'.

Field 8 The name of the limit switch mnemonic. The default is blank, indicating no limit switch is used. The limit switch is discussed below.

Field 9 A floating point value representing the limit switch low value. The default is '-DBL_MAX'.

Field 10 A floating point value representing the limit switch high value. The default is 'DBL_MAX'.

Field 11 The inversion flag; 'T' or 'F'. Discussed below; the default value is 'F'.

Field 12 The limit set description.

Limit definitions allow any number of limit sets for each mnemonic; however, currently, the ground system supports only two limit sets.

The limit switch (fields 8 - 10) is used to determine which set is to be used for limit checking. The raw (unconverted) value for the given "limit switch mnemonic" (field 8) (the name of any integer or floating-point mnemonic) is tested against the range presented by the limit switch low (field 9) and high (field 10) values as:

```
if (fld9_low != fld10_high) {
    if (fld9_low <= fld_8_mnemonic_value < fld10_high) {
        Use this limit set;
    } else {
        Keep on looking;
    }
} else { // fld9_low == fld10_high
    if (fld9_low == fld_8_mnemonic_value) {
        Use this limit set;
    } else {
        Keep on looking;
    }
}
```

The first set for which the limit switch condition is true is used for limit checking, but limit sets should not be allowed to overlap. If no limit switch conditions are true, the limit set provided without a limit switch is used. If no such set is provided, limits are not checked. Only one limit set without a limit switch will be accepted. Currently only one limit set with a limit switch will be accepted.

Limits are checked as follows:

```
if (red_high_limit_exists && value >= red_high_limit)
    if (invert) violation = RED_LOW;
    else        violation = RED_HIGH;
else if (yellow_high_limit_exist && value >= yellow_high_limit)
    if (invert) violation =  YELLOW_LOW;
    else        violation =  YELLOW_HIGH;
else if (red_low_limit_exists && value <= red_low_limit)
    if (invert) violation =  RED_HIGH;
    else        violation =  RED_LOW;
else if (yellow_low_limit_exists && value <= yellow_low_limit)
    if (invert) violation =  YELLOW_HIGH;
    else        violation =  YELLOW_LOW;
else  violation = WITHIN_LIMITS;   /* parameter in limits */
```

Note the use of the inversion flag (field 11) in the above code fragment. It serves transforms a high violation into a low violation.

Although not shown in this simplified code fragment, a mnemonic must be received in the same state of violation or in-limits twice consecutively before the ground system reports the limit condition.

4.6 The MAP Record

Packet attributes.

Field 1 'MAP', indicating the record type.

Field 2 The packet application ID. An integer value. Legal values are 0 to 65535 however CCSDS packets are limited from 0 to 2047.

Field 3 '+' or '-', indicating the operation symbol.

Field 4 List of subsystem names indicating which subsystems this packet is associated with. Up to 16 subsystem names may be specified using spaces to seperate the names.

Field 5 Packet timeout (a time value). Defaults to 0, which means that packet never times out. See *GBL_STATOUT* for the global timeout.

Field 6 Reserved for future use: *Packet length. Either blank, an integer value, or a mnemonic name. If it's a mnemonic name, there must be a corresponding PKT record that describes how to unpack the packet length.*

Field 7 Timestamp mnemonic. Either blank or a mnemonic name. If it's a mnemonic name, there must be a corresponding PKT record that describes how to unpack the timestamp. This identifies what PKT record if any is used to update *GBL_MR_PKTTIME* which is the time of last received packet. If blank no update is done. *If mnemonic name is followed by "NO_MR_PKTTIME", timestamp mnemonic will not update GBL_MR_PKTTIME. This is a special case for ITOS internal use to define a secondary header time field.*

Field 8 The packet's description.

4.7 The PKT Record

A telemetry mnemonic's value may be extracted from any telemetry packet. It may be present multiple times in a packet and in multiple packets. Each PKT record defines how to extract a single telemetry value from single occurrence in one packet.

Field 1 'PKT', indicating the record type.

Field 2 The packet application ID. An integer value. Legal values are 0 to 65535 however CCSDS packets are limited from 0 to 2047.

Field 3 A telemetry mnemonic name, as specified in Field 2 of a TLM record. Identifies the TLM mnemonic this record's data point unpacks into.

Field 4 The array index, an integer value used for mnemonic arrays the elements of which are not telemetered at constant intervals in a packet or are not in the same packet. Normally, array elements are in one packet at a constant distance from one another. The "offset between array elements" (field 11) gives this constant distance.

Field 5 '+' or '-', indicating the operation symbol.

Field 6 Unused.

Field 7 The "source type" of the mnemonic (see Section 2.6 [type codes], page 6). Indicates how the mnemonic's value is unpacked. If not specified, a default is derived from the mnemonic's TLM record's "destination type" (field 6).

Field 8 An integer value representing the "starting byte" where the data value begins, measured from the beginning of the packet primary header. In other words, the first byte in the packet primary header is byte 0. The starting byte plus the length of the packet item in bytes must not exceed the maximum packet length of 65529.

Field 9 An integer value representing the "starting bit" within the starting byte where the data begins. The most significant bit is bit 0. If blank, this will be assumed to be zero.

Field 10 The "field length", an integer value representing the number of bits comprising the data. If "source type" (field 7) is CHAR or S then length is in bytes. If blank, the length is inferred from the type, or is copied from the "size" (field 7) element of the mnemonic's TLM record. If source type is a TIME or DATE type then this field can be used to identify a telemetry mnemonic that defines the epoch used to decom the data. If omitted the default epoch is used. GBL_DEF_EPOCH can be used or any other DATE type mnemonic defined. The subseconds define how to convert from spacecraft subseconds to microseconds for both DATE and TIME mnemonics. If the telemetry mnemonic defining the epoch is an array with a size greater than one, then the epoch is adjustable (i.e. a different epoch will be applied whenever the specified reference time reaches one of the specified adjustment dates).

Note that the end bit given by ("start bit" + "length" - 1) may not fall outside the field selected by the source type. It is important to remember this rule when specifying a map for unpacking arrays.

Field 11 The "array offset", an integer value representing the offset in bits between array elements. Not used unless the mnemonic is an array. If blank, the default value comes from field 10 (the field length). If 0 only one array element is extracted.

Field 12 The selector definition name. This allows a reference to a set of SEL records.

Field 13 description.

4.8 The SEL Record

The SEL or SELECT record allows packet data to direct how part of a packet is to be unpacked by the use of multiple sub-packet definitions using a pseudo appid. This appid

is outside the normal range of appids coming from the spacecraft. This is also commonly called data directed unpacking.

WARNING - Caution should be taken not to cause an recursive loop situation where one packet includes a second packet which includes the first and so on.

Here's how they are intended to be used:

1. A PKT record points to the set of SEL records. Let's call the mnemonic in the PKT record the *selector*.
2. Sort the SEL records in ascending order using 'low bound' as the primary key and 'high bound' as the secondary key.
3. Loop through the sorted SEL records:
4. If 'low bound .LE. selector value .LE. high bound', then 'pseudo appid' identifies the map to use to unpack a portion of the packet.

Field 1 'SEL', indicating the record type.

Field 2 The "selector set name". Selector set names must begin with a letter, may contain only alphanumeric characters or underscore ('_'), and may be up to 255 characters long.

Field 3 '+' or '-', indicating the operation symbol.

Field 4 The "pseudo appid". An integer value. Legal values are 0 to 65535 but should be above the normal appids used by the spacecraft. For CCSDS this is a number above 2047.

Field 5 low bound. A floating point value.

Field 6 high bound. A floating point value.

Field 7 description.

5 Command Records

The CMD records define command mnemonics. Each command mnemonic maps to a particular ID and set of named command packet fields. In CCSDS commands, the ID is the application ID in the command packet primary header. Command data fields are defined in FLD records, and each identifies a set of contiguous bits in the command packet data field.

FLD records may contain a high-low range restricting the values to which the field may be set. Each field also may have associated with it a set of discrete values to which the field may be set. Defined in one or more SUB records, this set of values is the inverse of a telemetry mnemonic's discrete conversion: They provide a set of names each of which map to a fixed value for the field.

In order to understand the command database, it is necessary to understand how spacecraft commands are specified in the STOL language. The STOL syntax for a spacecraft command is:

```
["cmd " | /]<mnemonic> <submnemonic>[, <submnemonic>, ...]
```

for example:

```
/heaterctl shade, temp=22.4
/heateroff shade
```

The '`/`' indicates the following word is a spacecraft command. The second command could have been given alternatively, for example, as '`cmd heateroff shade`'. '`heaterctl`' and '`heateroff`' are command mnemonics. '`shade`' and '`temp`' are submnemonics; that is, parameters to the '`heaterctl`' or '`heateroff`' commands.

Notice the two forms of submnemonic presented. '`shade`' is often referred to as a "fixed" submnemonic, and '`temp`' is called a "variable" submnemonic. In the database, '`temp`' is the name of a command data field (FLD), and '`shade`' is one of the names from a set of discrete values associated with another field.

Note that it would have been possible to enter the command as:

```
/heateroff heater=shade
```

assuming that '`heater`' is the name of a field referencing a set of fixed values including one named '`shade`'. This syntax is required if the all the discrete value names don't map to a unique field for the given command. For example, for the command:

```
/set_relays on, off, on, on
```

it is not possible to tell which "on" or "off" refers to which field. This requires a syntax like:

```
/set_relays main=on, aux=off, inst=on, acs=on
```

The field names for each command must be unique for that command. Two or more commands can have fields with the same name, but no two fields within a single command can have the same name.

If a discrete value name is unique among all discrete value sets referenced by all fields in a single command, and among the field names for the command, then that name may be given alone in a STOL spacecraft command to specify both the field and the value to

which that field is being assigned. (Like "shade" in the examples.) Otherwise, the syntax '`<field name> = <discrete value name>`' must be used.

If a field definition contains a value range or does not reference a discrete value set, then the field value may always be specified to STOL in the "variable submnemonic" syntax. (Like '`temp`' in the examples.)

If a field definition references a discrete value set but does not contain a value range, then the field value must be set using one of the discrete value names. There will be a STOL mode to override this restriction for testing purposes.

Normally, STOL requires that a value be given explicitly for all fields. However, it is possible to set up the database so that commands may contain "hidden" or "optional" fields.

If an FLD record references a set of discrete values, and one value in the set is mapped to the name "default", then the "default" value is automatically used for the field if no other value is given for it in the STOL command. This is then an optional field, from the STOL perspective.

If a FLD record specifies a range where the low and high values are equal, so that the field can only take one value, then that value automatically will be supplied; there's no need to create a SUB record with the name "default" and the STOL user doesn't have to specify the [one and only] value for the field.

The following is an example database exerpt for the two commands "heaterctl" and "heateroff" used as examples above. Note that "heateroff" has the same command AppID and function code as "heaterctl", but the "temp" is set to a fixed, "hidden", value. These command might be defined as follows:

```
SSI,thermal,+,"Heaters and stuff"
CMD,heaterctl,+,1,1,thermal,,,80,,,,,,Command to control various heaters
FLD,heaterctl,heater,+,U1,,8,0,8,,,,heaters,Select heater to control
FLD,heaterctl,temp,+,F12345678,,9,0,64,,-10,98.6,,
Temperature to which heater should be set
SUB,heaters,shade,+,1,,Instrument shade
SUB,heaters,body,+,2,,Instrument body
SUB,heaters,detect,+,4,,Detector
SUB,heaters,all,+,7,,All heaters
CMD,heateroff,+,1,1,thermal,,,80,,,,,,Command to control various heaters
FLD,heateroff,heater,+,U1,,8,0,8,,,,heaters,Select heater to control
FLD,heateroff,temp,+,F12345678,,9,0,64,,-10.0,-10.0,noheat,
Temperature to which heater should be set
SUB,noheat,default,+,-10.0,,Temp setting to turn heaters off
```

The command records are

'`CMD`' define a command,

'`FLD`' define a command field, and

'`SUB`' define a discrete value for a command field.

5.1 CMD

Field 1 'CMD', indicating the record type.

Field 2 The "command mnemonic" is the name of the command and is how STOL procedures reference the command. Command mnemonic names must begin with a letter, may contain only alphanumeric characters or underscore ('_'), and may be up to 255 characters long.

Field 3 '+' or '-', indicating the operation symbol.

Field 4 This field is ignored for non-CCSDS commands.

The "application ID" and "default virtual channel" are combined in this field; values for this field are either integer value or two integer numbers seperated by 'VC'.

In the first form, where the field contains only one number, that number is the value for the CCSDS telecommand packet's primary header application ID field. Valid values are 0 to 2047. The default virtual channel is said to be unspecified.

In the second from, the first number is the application ID and the second number is the default virtual channel in the range of 0 to 63. **This form is not currently supported but reserved for future use:**

The default virtual channel is used in command processing: unless overridden in STOL's CMD directive, the command will get transmitted on the default virtual channel. If the default virtual channel is unspecified, the command is transmitted on the virtual channel identified by GBL_VC.

It is expected that most commands will leave the default virtual channel unspecified.

Field 5 The "function code" (for SMEX commands) or command type indicator.

This is one of:

CCSDS This is a CCSDS command.

RAW This is a non-CCSDS command.

15-bit unsigned integer (deprecated)
: This is a SMEX variant of a CCSDS command. The value given is the value for the CCSDS telecommand packet secondary header. The function code's placement in the packet is defined by the SH_FUN_CODE which is part of the ITOS contribution to the database, defined in 'itos/dbx/cmdhdr.dbx'.

Field 6 List of subsystem names (see Section 3.2 [ssi], page 13) indicating which subsystems this command is associated with. Up to 16 subsystem names may be specified using spaces to seperate the names.

Field 7 The "dump flag" is one of the characters:

' ' (blank, the default)
: This is not a dump command.

'A' This command causes the spacecraft to abort the current dump.

'T' This command causes the spacecraft to initiate a table dump.

'M' This command causes the spacecraft to initiate a memory dump.

Field 8 The "telemetry AppID for dump data" is used only for dump commands. It gives the application ID of the telemetry packet containing the requested dump data. This field with the "dump flag" allows the ground system to automatically capture spacecraft dumps, and to recognize when such a capture should be aborted.

Like the "application ID" field above, this is actually a combination of application ID and virtual channel; "7 VC 1" indicates appid 7 on vc 1. If the VC option is not specified, VC 0 is assumed otherwise it must be in the range of 0 to 63.

Field 9 The "command length in bits". ("In bits" rather than "in bytes" is a historical artifact; the length in bits is always a multiple of eight!). In most cases, you should leave this field blank, in which case `dbxodb` will supply the minimum command length that will hold all of the fields specified for the command.

For CCSDS commands this is the value of the primary packet header's "packet length" field multiplied by 8; that is, this is the length in bits of the application data field including any secondary header, minus 8 bits.

For example, consider a command packet that is 15 bytes total (including **all** headers): Since the packet primary header is 6 bytes long, the application data field (plus secondary header) is 9 bytes long. So the "command length in bits" is 9 bytes times 8 bits per byte minus 8 bits, or 64 bits.

For non-CCSDS commands, this is the length of the whole command packets. So the "command length in bits" of a 15-byte non-CCSDS command is 15 bytes times 8 bits per byte, or 120.

The length entered here should **not** include the length of any checksum specified in this command record or through GBL_CHKSMROUT.

Field 10 The "run-time flag" is a historical artifact that indicates the context in which the command is valid: real-time, in an absolute time sequence (ATS), and/or a relative time sequence (RTS). Its values include:

0 real-time, RTS, and ATS (default)

1 real-time only

2 RTS only

3 ATS only

4 real-time and RTS

5 real-time and ATS

6 RTS and ATS

7 CMS (command management system) only

Field 11 The "criticality flag" indicates that the ground system should prompt for permission to send the command whenever the command is issued. This field may be set to one of:

H	hazardous
R	critical
C	conditionally critical
Z	conditionally hazardous
N	not critical (default)

If flag is '`C`' or '`Z`' see field 12. Any flag other than '`N`' will cause a POPUP window to open and require the operator to authorize the release of the command. Hazardous commands require a pass phrase to also be entered in the POPUP window that is compared against the global mnemonic variable *GBL_STOLHAZ_PHRASE*.

Field 12 The "critical condition string" contains a STOL logical expression evaluating to true/false. If the "criticality flag" in field 11 is not set to '`C`' or '`Z`', this field is ignored. If the expression evaluates as true when the command is issued, the command is treated as a critical or hazardous command as described in field 11. The maximum expression length is 255 characters and must be enclosed in double quotes. Parentheses are optional except when required for expression order of evaluation. For example:

```
"(HKTOISRCNT .GT. 5 .AND. HKCRESETCNT .EQ. 0)"
```

Field 13 Some commands require special checksums. The "checksum function name" is the name of an checksum algorithm to be applied to the command. The maximum length is 255 characters.

Field 14 Reserved for future use: *The "end-item verification string" contains a a STOL logical expression. STOL will wait for the expression to become true after the command is sent to the spacecraft before allowing the next proc statement to be executed . The maximum length of this string is 255 characters. It's contents will be parsed but ignored at this time.*

Field 15 description.

5.2 FLD

Field 1 '`FLD`', indicating the record type.

Field 2 Command mnemonic names must begin with a letter, may contain only alphanumeric characters or underscore ('_'), and may be up to 255 characters long.

Field 3 field name (up to 255 chars). The "field name" is a descriptive name for the field.

Field 4 '`+`' or '`-`', indicating the operation symbol.

Field 5 The "destination data type" is analogous to the "source data type" in the PKT record. It is the field's data type (see Section 2.6 [type codes], page 6) which indicates the type of value for the field (unsigned integer, IEEE floating point. string, date, etc) and byte order of the value.

Field 6 The "array size" gives the array length if this record defines an array field. Reserved for future use.

Field 7 The "starting byte" is the byte where the field begins; byte 0 is the first byte of the command packet. I.e., for CCSDS commands, byte 0 is the first byte of the packet primary header. Except for *GBL_LCLHDR* special commands, the starting byte of a command field for MUST be greater than 5 for CCSDS commands and 7 for SMEX commands or it will be flagged as an error.

Field 8 The "starting bit" is the bit within the destination field where the data field begins. Bits are numbered big endian starting with 0; the high order bit of each byte is bit 0. Note that specifying a non-zero starting bit only makes sense for signed or unsigned integral values. If blank, this will be assumed to be zero.

Field 9 The "field length".

For signed or unsigned integral data types, this is the length in bits and must be less than or equal to the maximum length for the type. If not specified, the default is the maximum length for the type, i.e. 8 for UI, 16 for U12, etc.

For string data types, this is the maximum length of the string. If the field length is 5, the strings "1", "12", "123", "1234", "12345" will fit in the field but the string "123456" won't. If not specified, the default is 1. Strings that are shorter that the maximum length will be left justified within the field. Note that this does not provide for any Nil (zero) character to mark the end of the string.

For TIME or DATE types, this field can be used to identify a telemetry mnemonic that defines the epoch used to encode the data. If omitted the default epoch is used. GBL_DEF_EPOCH can be used or any other DATE type mnemonic defined. The subseconds define how to convert from microseconds to spacecraft subseconds for both DATE and TIME fields.

For all other data types, this field should be left blank.

Note that for arrays, this is the length of each element, not of the whole array.

Field 10 The "offset between array elements" is the distance between the start of a data element in an array field and the start of the next element in the array. This normally is in bytes, but may be given in bits by appending a "b" (either case) to the number. This field is ignored if the field is not an array; that is, if "array size" (field 6) is zero, one, or blank.

Field 11 field value range lower bound

Field 12 field value range upper bound

The "field value range lower" and "upper" bounds, provide a range of valid values for the field. **Ranges are not used with string fields.** The field value must satisfy the expression:

```
lower_bound <= field_value <= upper_bound
```

Note that field values are automatically bounded somewhat by data type and and field length, such that a 2-bit unsigned field is automatically constrained to be valued from 0 to 3, for example. So, if no range is specified, the value will be checked only to see that it fits in the given field length. If no SUB records are defined for this field and lower and upper bounds are the same value then that value will act the same as the "default". If so, STOL will not allow a value for this field making it invisible or hidden.

Field 13 The "discrete value set name" gives the name of a discrete value set containing a set of values to which the field may be set. As noted in the commentary at the head of this section, this field and the preceding two fields (the value range) interact to determine how the field value may be set.

Field 14 description.

5.3 SUB

Field 1 'SUB', indicating the record type.

Field 2 The "discrete value set name" is the name used in field 13 of the FLD record. It names a set of SUB records. This may be up to 255 characters long.

Field 3 The "value name"is the name for this discrete value in the set. This is the name used in STOL when sending the command to specify the associated value for some field. This may be up to 255 characters long.

The special name "default" marks this as the field's default value, i.e. the value the field will be set to if no other value is specified.

Field 4 '+' or '-', indicating the operation symbol.

Field 5 The "fixed value" is the fixed value associated with "value name". If this value is associated with a FLD record of string type (S1 or S21) the maximum string length of the fixed value is 255 characters.

Field 6 Conditionaly critical or hazardous submnemonic flag. The "criticality flag" at this level allows some field values to be considered critical while others are not. This field only has value if the criticality flag in field 11 of the CMD record is 'N'. If a command is sent and a criticality value is specified, the command is treated as critical. Command field discrete values may not be marked conditionally critical. Valid values are:

H	hazardous
R or Y	critical
N	not critical (default)

'Y' is for backward compatiblity but is deprecated in favor of 'R' and may not be supported in the future. Any flag other than "N" will cause a POPUP window to open and require the operator to authorize the release of the command.

Field 7 description.

6 Special commands and mnemonics

6.1 Special mnemonics

The following mnemonics must be defined. If missing dbxodb will create them with specific initial values!

```
GBL_MISSION
               TLM,GBL_MISSION,+,,,S,32,,,,,,"trace",,
                   "The mission this database is for."

GBL_DBVERS
               TLM,GBL_DBVERS ,+,,,S,32,,,,,,"4.1",,
                   "Database version."

GBL_DEF_EPOCH
               TLM,GBL_DEF_EPOCH,+,,ITOS_DB,TIME,,,,1,,,68-145-0:0:0.065536,F,
                  "Default EPOCH time adjustment. GBL_DEF_EPOCH is the date of the
                  spacecraft epoch. The default initial value is 'Midnight May 24,1968'
                  the SMEX epoch.
                  The factional part is the conversion denominator to convert from
                  spacecraft subseconds to microseconds, 065535 is default for SMEX.
                  The leading zero is needed to account for a 6 digit number.

                  SMEX  Initial Value: '68-145-0:0:0.065536'
                  Swift Initial Value: '68-145-0:0:0.050000'
```

6.2 Special commands

The following commands must be defined:

```
GBL_LCLHDR
               CMD|GBL_LCLHDR|+|||ITOS_CMD|||||||||
                  "CCSDS Command Local Header Fields"
               FLD|GBL_LCLHDR|PH_SEC_HDR|+|UB||0|4|1||1|1||
                  "The CCSDS packet header Secondary Header Flag.  Must be
                  1 to indicate that a secondary header is present."

               FLD|GBL_LCLHDR|PH_APPID|+|UI||0|5|11|||||
                  "The CCSDS packet header Application Process Identifier (APPID)."

               FLD|GBL_LCLHDR|PH_PKT_LEN|+|UI||4|0|16|||||
                  "The CCSDS packet header Packet Length.  CCSDS defines this as the
                  number of octets in the packet minus 1."

               FLD|GBL_LCLHDR |SH_FUN_CODE|+|UI||6|1|15|||||
                  "Command Function Code."
```

7 Time Adjustments

Adjustable epochs are used to handle time discontinuities such as leap seconds and clock rollover. For packet items that require time adjustments, the name of the mnemonic defining the adjustable epoch array must be specified in the PKT record's field 10. In all, four parallel arrays must be defined including the aforementioned epoch array, a reference date mnemonic array used to determine if the epoch needs to be adjusted, an adjustment date array indicating the dates and times when the epoch must be adjusted, and a reference near zero array used to determine when the reference date has reached the adjustment date. Additional mnemonics that must be defined include an absolute target near zero time that indicates when the target date has gone back to zero and the number of adjustments to be made.

The adjustment dates must be given as absolute times and ordered by time. Each adjustment date has an associated reference date that can be the target date itself (specified using the keywork '`SELF`') or some other absolute time mnemonic, such as ground received time. '`SELF`' references are used for fine adjustments such as leap seconds. For coarse adjustments, such as clock rollover, some other reference must be used. The epoch, reference, and near-zero value is associated with each adjustment date. Adjustment dates are compared with the reference using the near-zero value. For each adjustment date, if the reference date is less than the adjustment date, then that adjustment date's corresponding epoch is selected. Coarse adjustments are made first, then fine adjustments. Logic is included to handle boundary conditions for coarse adjustments. The target-near-zero value is used to determine if a date value is near zero to test for rollover.

The names of the global mnemonics and arrays must adhere to the following naming conventions, where the base name begins with the name of the mnemonic defining the adjustable epochs:

number of adjustments
: base name concatenated with "_adj_count"

adjustment date array
: base name concatenated with "_adj_date"

reference mnemonic array
: base name concatenated with "_ref_date"

reference near zero time array
: base name concatenated with "_ref_near_zero"

target near zero date
: base name concatenated with "_target_near_zero"

For example, if the epoch mnemonic name is '`gbl_adjepoch`', the following tlm records would be used to create the associated global mnemonics:

```
tlm,gbl_adjepoch,+,,itos_test,time42,,,,10,,,
  01-001-00:00:00.065536,F,"adjustable epoch."
tlm,gbl_adjepoch_adj_date,+,,itos_test,time42,,,,
  10,,,01-001-00:00:00.065535,F,
  "adjustable epoch adjustment dates array."
```

```
tlm,gbl_adjepoch_ref_near_zero,+,,itos_test,UI,,,,10,,,30,F,
  "adjustable epoch reference near zero array."
tlm,gbl_adjepoch_target_near_zero,+,,itos_test,time42,,,,,,,
  06-001-00:00:20.0,F,"adjustable epoch target near zero."
tlm,gbl_adjepoch_ref_date,+,,itos_test,CHAR,30,,,10,,,SELF,F,
  "adjustable epoch reference date mnemonic array"
tlm,gbl_adjepoch_adj_count,+,,itos_test,UI,,,,,,,5,F,
  "number of adjustment dates"
```

The global mnemonic values are assigned at system startup by including the settings in the itosrc file. For example, the following entries for the mnemonics defined above might appear in the itosrc file as shown below:

```
# Time Adjustment Globals
setenv ITOS_ADJEPOCH_TARGET_NEAR_ZERO 06-001-00:00:05.0
setenv ITOS_ADJEPOCH_ADJ_COUNT 6
# TIME ADJUSTMENTS: ADJUSTMENT DATE ARRAY
setenv ITOS_ADJEPOCH_ADJ_DATE_0_ 06-001-00:00:30.0
setenv ITOS_ADJEPOCH_ADJ_DATE_1_ 06-001-00:00:40.0
setenv ITOS_ADJEPOCH_ADJ_DATE_2_ 06-001-00:01:10.0
setenv ITOS_ADJEPOCH_ADJ_DATE_3_ 07-001-00:00:00.0
setenv ITOS_ADJEPOCH_ADJ_DATE_4_ 06-001-00:00:50.0
setenv ITOS_ADJEPOCH_ADJ_DATE_5_ 07-001-00:00:00.0
# TIME ADJUSTMENTS: REFERENCE NEAR ZERO ARRAY
setenv ITOS_ADJEPOCH_REF_NEAR_ZERO_0_  1
setenv ITOS_ADJEPOCH_REF_NEAR_ZERO_1_  1
setenv ITOS_ADJEPOCH_REF_NEAR_ZERO_2_  1
setenv ITOS_ADJEPOCH_REF_NEAR_ZERO_3_  1
setenv ITOS_ADJEPOCH_REF_NEAR_ZERO_4_  5
setenv ITOS_ADJEPOCH_REF_NEAR_ZERO_5_ 10
# TIME ADJUSTMENTS: REFERENCE DATE ARRAY
setenv ITOS_ADJEPOCH_REF_DATE_0_ "SELF"
setenv ITOS_ADJEPOCH_REF_DATE_1_ "SELF"
setenv ITOS_ADJEPOCH_REF_DATE_2_ "SELF"
setenv ITOS_ADJEPOCH_REF_DATE_3_ "SELF"
setenv ITOS_ADJEPOCH_REF_DATE_4_ "ITOS_REF_TIME"
setenv ITOS_ADJEPOCH_REF_DATE_5_ "ITOS_REF_TIME"
# TIME ADJUSTMENTS: EPOCH ARRAY
setenv ITOS_ADJEPOCH_0_ 01-001-00:00:00.065536
setenv ITOS_ADJEPOCH_1_ 01-001-00:00:05.065536
setenv ITOS_ADJEPOCH_2_ 01-001-00:00:10.065536
setenv ITOS_ADJEPOCH_3_ 01-001-00:00:15.065536
setenv ITOS_ADJEPOCH_4_ 01-001-00:00:00.065536
setenv ITOS_ADJEPOCH_5_ 01-001-00:00:51.065536
```

8 The dbxodb program

The dbxodb program processes these transaction records and produces any combination of

- a new operational database
- a set of PDB files (for use by the CMS & DTAS systems)
- a T&C volume II report in HTML
- a T&C volume II report suitable for printing

```
$ dbxodb -help
Usage: dbxodb <options> <files containing dbx transactions>
where <options> are:
  -mkodb, -noodb, -mkhtml, -nohtml, -mkvII, -novII, -mkpdb, -nopdb,
  -mkdbx, and -nodbx
      -- these control which outputs get produced.  The defaults are
         -mkodb, -nohtml, -novII, -nopdb, and -nodbx.
  -odbdir <path>  -- turns on -mkodb and specifies where ODB files get
         written.  Defaults to the current directory.
  -htmldir <path> -- turns on -mkhtml and specifies where html files get
         written.  Defaults to the current directory.
  -urlbase <url>  -- turns on -mkhtml and specifies the URL to the directory
         where html files get written.  If specified, the generated files
         have a <BASE href=...> specification in their <HEAD>.
  -vIIfile <path> -- turns on -mkvII and specifies the basename of the
         output files.  Defaults to './<mission>.<dbvers>.tcvII'; the four
         output files have suffixes '.tp', '.mnem', '.pkt', and '.cmd'.
  -pdbdir <path>  -- turns on -mkpdb and specifies where the PDB files get
         written.  Defaults to the current directory.
  -oldpdb -- PDB files generated in pre TRACE era format.
  -dtas   -- PDB files are generated with 84 char records using
         16 character mnemonic names instead of default 12
         used by the Data Trending & Analysis System (DTAS).
  -dbxfile <path> -- turns on -mkdbx and specifies where the dbx transactions
         get written.  Defaults to './<mission>.<dbvers>.yymmddhhmmss.dbx'
  -verbose -- turns on additional output messages.  '-verbose -verbose'
         makes things even more verbose!
  -silent  -- turns off everything except error messages.  The default,
         error and warning messages, is equivalent to '-silent -verbose'.
  -mission <mission> -- Names the mission this database is for.
  -version <version> -- Identifies the database version.
  -epochdef <mnemonic name> -- Use <mnemonic name> as the default epoch
         conversion mnemonic on all PKT and FLD records of time and
         date type that don't have an epoch mnemonic defined.
         Default is no default epoch mnemonic is used, and ITOS
         telemetry command subsystems and will use the SMEX epoch of
         00:00:00 May 24, 1968.
Special options to enlarge the database for realtime additions are:
  -maxname <number> -- Max mnemonic, limit, conversion, cmd ... names size,
```

```
       Defaults to 16.",
  -maxstr <number> -- Max size of strings, descriptions, etc. Defaults to 32.
  -maxmne <number> -- Number of extra mnemonics.
  -maxmap <number> -- Number of extra packets.
  -maxsel <number> -- Number of extra SEL records.
  -maxpkt <number> -- Number of pkt items per packet. Defaults to 25.
  -maxpksz <number> -- Max packet size in bytes for frame sim. Defaults to 1790
  -maxlim <number> -- Number of extra limit sets.
  -maxdsc <number> -- Number of extra discrete conversion sets.
  -maxdsi <number> -- Number of discrete items per conversion set. Default is 25
  -maxalg <number> -- Number of extra analog conversion sets.
  -maxxpr <number> -- Number of extra expression conversion sets.
  -maxxpi <number> -- Number of expression items per conversion set.
       Default is 4096 (12 bits).
  -maxcmd <number> -- Number of extra cmds.
  -maxfld <number> -- Number of extra fields per command. Default is 10.
  -maxsub <number> -- Number of extra subfields per command. Default is 25.
When conflicting options are specified, the last specification is used.
```

A Common Database Build Scenario:

```
$ dbxodb -mission trace -version 4.11 -odbdir /home/trace/odb/sparc_SunOS \
-htmldir /home/trace/public_html/tcvol2 \
/home/itos/dbx/*.dbx /home/trace/dbx/trace.sdb4.11
```

`-mission trace` and `-version 4.11` override any initial value in the TLM record for the *GBL_MISSION* and *GBL_DBVERS* mnemonics.

`-odbdir /home/trace/odb/sparc_SunOS` names the ODB directory. The following files will be created:

```
/home/trace/odb/sparc_SunOS/trace.odb4.11
/home/trace/odb/sparc_SunOS/trace.subsys4.11
/home/trace/odb/sparc_SunOS/trace.tags4.11
/home/trace/odb/sparc_SunOS/trace_cmd.odb4.11
/home/trace/odb/sparc_SunOS/trace_simbuf.odb4.11
/home/trace/odb/sparc_SunOS/traceavlcmdmne4.11
/home/trace/odb/sparc_SunOS/traceavlmnem4.11
/home/trace/odb/sparc_SunOS/traceavlmnemid4.11
/home/trace/odb/sparc_SunOS/traceavlpackid4.11
```

`-htmldir /home/trace/public_html/tcvol2` names the HTML directory.

The remaining parameters identify the input transaction files. The `/home/itos/dbx/*.dbx` files are the ITOS-provided transaction files that define ITOS global mnemonics, command headers, etc. The transaction files are parsed in the order in which they are included on the `DBXODB` command line. If there are duplicates, the last one encountered takes prescedence.

In addition to the above, DBXODB adds a set of global mnemonics "GBL_PKTCNT_xxxx" where "xxxx" is a 4 digit packet id number for all the packets defined with PKT records. These mnemonics are used by the pktcount display *PAGE*. The user can add records for packets that may be received but have no PKT

records so that the pktcount page will display these counts. For example creating a TLM record:

```
TLM|GBL_PKTCNT_0300 |+||ITOS_TEST|U1234|32|||||||T|"pkt 300 receive count"
```

would add a mnemonic for GBL_PKTCNT_0300 which will cause pktcount to display the counts for packet 300. Legal packet ids created by the user may be between 0000 and 2045.

Index

P

R

S

T

U

X

Y

Table of Contents